Delay-Tolerant Collaborative Filtering

Patrick Gratz
University of Luxembourg
6, rue Coudenhove-Kalergi, L-1359 Kirchberg,
Luxembourg
{patrick.gratz}@uni.lu

Tom Leclerc
MADYNES - INRIA, Nancy Grand-Est, France
Campus Scientifique - BP 239 54506
Vandœuvre-les-Nancy Cedex
{tom.leclerc}@loria.fr

ABSTRACT

Recommender systems using collaborative filtering are a well-established technique to overcome information overload in today's digital society. Currently, predominant collaborative filtering systems mostly depend on huge centralized databases to store user preferences and furthermore are only available when connected to Internet. In this paper, we consider an incremental recommender system for highly dynamic mobile environments where no central global knowledge is available and communication links are rather unreliable in comparison to static networks. We present an algorithm that aims to reach a reasonable prediction coverage and accuracy while keeping the amount of additional network overhead as small as possible, maximizing the performance of our system. For this purpose, the presented algorithm is based on a delay-tolerant broadcasting mechanism on top of a weighted cluster topology. Evaluation results show that in terms of accuracy and coverage the results of the presented algorithm converge on those obtained from a global knowledge scenario, even in the case of message loss.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Algorithms

Keywords

Delay-tolerant communication, mobile ad hoc networks, clustering, collaborative filtering

MobiWac'09, October 26–27, 2009, Tenerife, Canary Islands, Spain.

1. INTRODUCTION

Due to the ever increasing amount of available information in today's digital society, it becomes more and more difficult to determine the most relevant or useful information. Recommender systems using collaborative filtering (CF) are a well-established technique to overcome this problem of information overload by recommending information items based on taste of like-minded users [10]. Currently predominant systems using CF mostly depend on huge centralized databases to store user preferences and furthermore are only available online.

Consider the increase in popularity of mobile devices in the form of mobile phones, smart phones, PDAs and Tablet-PCs the same problem of information overload emerge. Different from traditional static devices such mobile devices often have no always available cellular connection to the Internet. Furthermore, compared to the often available wireless communication capabilities such a cellular connection cause more cost and provides lesser bandwidth.

In [7] we introduced a collaborative filtering system which enables the user to get podcast recommendations based on the taste of other like-minded users from nearby mobile environments. The described approach provides two different algorithms to determine similar neighbors in order to incrementally buid up a local model. Based on this local model the system can calculate rating predictions even in settings where no communication link is available. In this work, we present a delay-tolerant approach that aims to reach a reasonable performance concerning accuracy and coverage even under lossy settings, while keeping the amount of additional network overhead as small as possible. Although, the presented algorithm is based on a broadcasting mechanism on top of a stable weighted cluster topology.

The remainder of this paper is organized as follows. Related work is presented in Section 2. Section 3 briefly introduces a suitable similarity measure as well as the underlying cluster topology and presents a delay-tolerant intra-cluster broadcast in order to determine and disseminate similar neighbor profiles. In Section 4 we evaluate our system and compare the introduced algorithm with an idealized scenario and a best effort approach concerning performance and message complexity. Finally, our paper concludes with a preview to future work in Section 5.

2. RELATED WORK

Recommender systems using CF are a well established technique to reduce information overload. In the following we introduce some existing research work about CF based

recommender systems in mobile environments.
In [4] an incremental collaborative Filtering algorithm for applications, where users are occasionally connected to a central server is introduced. The general idea is to store a subset of selected user profiles, together with a ranked list of predictions. When the user is in offline mode, a service on the local device can still recommend items based on the predictions made the last time the user was connected. Each time the user supplies new ratings, the list of predictions will be recomputed, even if the user is not connected to the server. In the case that a user encounters another user, the authors suggest that they exchange their profiles and recalculate their prediction lists. The past influence of the other user should be removed from all predictions and the new influences should be added. At last this case is not evaluated or considered any further in the paper and is a part of future work.
A further portable recommender system along with five peer-to-peer (P2P) architectures for finding neighbors is presented in [9]. The authors introduce a new collaborative filtering algorithm called PocketLens that can run on connected servers, on usually connected workstations or occasionally connected portable devices. The presented algorithm is a variant of the item-item algorithm introduced in [10] with modifications for a peer-to-peer environment. To reach the goal of portability a local similarity model is created for the user. Thereby, the algorithm only needs access to the ratings of the owner and one other user at a time. In this manner, the model is created incrementally in a distributed fashion.
An approach to collaborative filtering in a mobile tourist information system for visitors of a festival based on spatio-temporal proximity in social contexts is proposed in [5]. This new approach is based on the idea that users who go to the same place at the same time tend to have similar tastes. In order to keep track about the visited places each user is equipped with a portable computer coupled with a GPS unit. Furthermore, a central server provides a database with information about all the events, restaurants, venues and bars at the festival [2]. The proposed approach uses a user-based CF technique and calculates similar users via a spatio-temporal proximity measure, i.e. two users are considered as similar if they consume the same items simultaneously. The following exchange of rating information between such similar users is done via an ad-hoc peer-to-peer interaction. However, the defined similarity measure has one drawback. Users consuming the same periodic event at different times still share interests, but are not considered as similar. In a future work, the authors intend to investigate how their CF approach can be extended in order to exchange ratings between users in spatial but not temporal proximity. Furthermore, they want to evaluate the introduced CF system at the Edinburgh Fringe festival.
In [11] an approach based on epidemic spreading of user preferences is presented and also evaluated. In their study the authors show how to reach a prediction accuracy in a mobile ad hoc scenario that is comparable to the prediction accuracy obtained in a global knowledge scenario. However, the presented evaluation lacks from several shortcomings. Firstly, no realistic mobility model has been used, but an idealized data exchange pattern with disjoint pairs per iterations. Furthermore, typical problems of mobile ad hoc networks, like unreliable connections affected by interferences or collision are not considered.

3. A DELAY-TOLERANT CF

Based on the work introduced in [7], our collaborative filtering system can be roughly divided into three phases: (1) requesting similar neighbor information, (2) updating the local recommender model and (3) a subsequent prediction calculation based on this local model. In addition, on each cluster-head there are two preceding phases: (p1) the retrieval of unknown profile information and (p2) the subsequent aggregation and provision of similar neighbor profiles. In the following section we introduce a suitable measure to determine the similarity between two user profiles and describe how the underlying cluster topology is created. Afterwards, we present our delay-tolerant intra-cluster based broadcasting mechanism for the retrieval, aggregation and provision of similar neighbor information.

3.1 User Similarity

In order to calculate the similarity w between an active user a and a neigbhor u we used the Pearson correlation coefficient which is defined as follows:

$$w_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \bar{r}_a) * (r_{u,i} - \bar{r}_u)}{\sigma_a * \sigma_u}.$$

Where $r_{u,i}$ is the rating for item i given by user u, $\bar{r}$ is the average rating and σ the covariance. However, one of the issue of this metric is that highly correlated neighbors are often based on a tiny number of co-rated items. To overcome this issue we applied a correlation significance weighting as introduced and discussed in [8].

3.2 Clustering Algorithm

Our algorithm is based on stable weighted cluster topologies generated by the Node and Link Weighted Clustering Algorithm (NLWCA) [1]. The clusters are formed by electing among the stable neighbors the node with the highest node weight. In order to generate and protect stable cluster structures, this algorithm also assigns weights to the links between the own node and the network neighbor nodes. This weight is used to keep track of the connection stability to the one-hop network neighbors. For neighbors staying in communication range this link weight is increased. When a link weight reaches a configurable stability threshold the corresponding network devices are considered to be stable connected. A cluster-head is elected only from the set of stable neighbors which avoids the re-organization of the topology when two clusters are crossing for a short period of time (Figure 1).

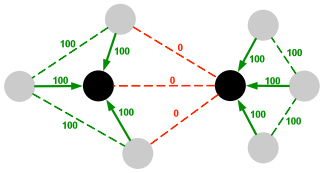

Figure 1: The low weight of the links avoids superfluous re-organization of the topology when for instance two clusters cross in mobile networks.

As the election of the cluster-head is always, among the stable neighbors, the node with the highest weight, sub-head

nodes can be formed. A sub-head is a node that elects a neighbor node as cluster-head but at the same time is elected as cluster-head by some other one-hop neighbor nodes. This can lead to a topological chain of sub-heads (Figure 2).

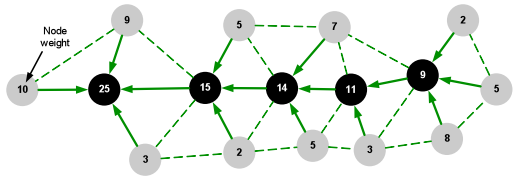

Figure 2: NLWCA cluster with a sub-head chain.

Sub-head chains can lead to a long multi-hop path inside one cluster. Long chains inside a single cluster increase the complexity of information exchange. A sub-head at the end of the chain has to synchronize with its cluster-head on the other side of the chain. The scalability advantage of clusters can be lost if long chains of sub-heads occur. Therefore, we propose an additional simple rule to the NLWCA cluster-head election: a node that already elected a foreign node as cluster-head is not eligible to be elected by another node as cluster-head. As a result if a node would elect the highest neighbor node which already elected a foreign cluster-head, the next highest neighbor node that hasn't elected a foreign cluster-head is elected as cluster-head. For illustration with the sub-head prevention rule the big cluster formed in Figure 2 would result in 3 clusters (nodes with weight 25, 14 and 9) shown in Figure 3. Here, node 3 cannot select node 11 as cluster-head and therefore picks the next highest neighbor which is node 9.

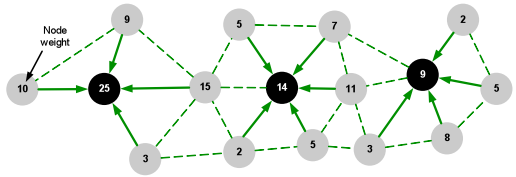

Figure 3: NLWCA cluster without a sub-heads.

3.3 Delay-Tolerant Intra-Cluster Broadcast

On top of the above mentioned topology our algorithm works as follows. Each cluster-head communicates exclusively with its stable slaves assigned by the described clustering algorithm. In order to detect and exchange only relevant profiles, additional beacon information is used. For this purpose, we extended the beacon of a slave device by the following information: a time stamp ($pTime$) with respect to the current profile information, the similarity of the least similar neighbor (LSN) stored in the local recommender model and an identifier of the last received cluster-head message (mID). If a cluster-head detects a new or updated profile of a slave device, he initiates a request procedure (Listing 1) to get the corresponding information. In order to avoid sending separate request messages to subsequently detected devices, the request procedure uses a delayed broadcast mechanism. Thus, each time a cluster-head initiates a request to a detected device, a specified timeout period is started. If during this timeout, further relevant devices are detected, the timeout period will be restarted and the corresponding devices will be noticed in a request-list. Otherwise (once the timer elapsed successfully), the cluster-head sends a broadcast containing the request-list.

Listing 1: Request procedure (pseudo code).

```
// request procedure
foreach received beacon from a stable slave
{  addr = slaveAddress;
   time = beacon.pTime;
   lsn = beacon.LSN;
   if(profileCache.contains(addr, time)) {
   lsnList.add(addr,lsn);
      if(updateID == beacon.mID)
         consistTable.set(addr,0); }
   else {
   if(requestTimer is running)
         requestTimer.reset();
      requestList.add(addr);
      requestTimer.start(); } }
foreach elapsed requestTimer {
   reqM = new requestMessage();
   reqM.addReceivers(requestList);
   sendBroadCast(reqM); }
```

After receiving a request from the elected cluster-head, the slave devices contained in the request-list start to send their current profile information to their cluster-head. In order to avoid simultaneous responses, each slave waits a random timeout before sending the profile. After receiving the first requested profile information, the cluster-head initiates an update procedure. As shown in Listing 2, the procedure starts with a further timeout phase and each received profile during this phase causes a restart of the timer.

Listing 2: Update procedure (pseudo code).

```
// update procedure
foreach received profile from a stable slave
{  addr = slaveAddress;
   requestList.remove(addr);
   profileCache.add(addr, profile);
   consistTable.set(addr,k+1);
   if(updateTimer is running)
      updateTimer.reset;
   updateTimer.start(); }
foreach elapsed updateTimer
{  profiles = recently received profiles;
   matrix.update(profiles);
   currentDelta = matrix.getDelta();
   consistTable.add(currentDelta);
   update = consistTable.getUpdate();
   updateID = update.getID();
   sendBroadCast(update);
   consistTable.increaseLevel(); }
```

Once the timeout period has successfully elapsed the cluster-head starts to calculate the corresponding profile similarities and manages the resulting values in a similarity matrix. Subsequently, the cluster-head aggregates relevant similar neighbor information for each slave and broadcasts the result to the cluster participants. Because of the limited storage capacity and in order to avoid the dissemination of non-relevant information, for each slave the cluster-head considers only neighbor profiles with a similarity value higher than the LSN from the recently received beacon. Once the recommender model at a slave device reaches its maximum capacity, this value is set to the similarity value of the least

similar neighbor stored in the local model, otherwise the value is set to -1.

k+1	k	...	1	0	Level
S_c	ΔS_{n-k}	...	ΔS_{n-1}	ΔS_n	Cache

Figure 4: Consistency table at the cluster-head.

Due to possible communication errors it its very likely that some devices are temporarily not able to receive updates. For this reason, each cluster-head caches a limited number of recently sent updates and maps each one-hop member to a corresponding consistency level (Figure 4) depending on whether he received an according acknowledgment (mID) via a beacon of this slave device or not. In the case, that there are one-hop members at different consistency levels the cluster-head calculates and broadcasts the least common update.

4. EXPERIMENTS AND RESULTS

In order to evaluate our algorithm we implemented it on top of JANE [6] and performed several experiments via an existing rating database. The following sub-sections describe the used data set, the evaluation criteria as well as the results under certain simulation settings.

4.1 Dataset

We used the MovieLens[1] Data Set that consists of 100,000 ratings for 1682 movies by 943 users, where each user has at least rated 20 movies. For our experiments, the data has been split into 5 different training and test sets for a five-fold cross validation. Each training set contains 80 % of the data and each test set the remaining 20 %, where each set maintains part of the ratings for both users and items.

4.2 Evaluation Criteria

In our experiments we used the Mean Absolute Error (MAE) metric as the criterion to evaluate the accuracy of the calculated predictions. Thus, after each simulation run we compare the predicted votes with the corresponding votes in the test set and calculate the MAE, as follows:

$$MAE = \frac{\sum_{i=0}^{M} |pred(i) - r(i)|}{|M|}.$$

If a predicted item did not have an adequate entry in the test set it was eliminated from the evaluation. In the case that there is no corresponding prediction for an item in the test set, we used the average user rating as prediction value. Note that we used the MAE only to compare how accurately our algorithms predict a randomly selected item rather than evaluating the user experience of generated recommendations. As second criterion we measured the predicition coverage. We compute the coverage as the percentage of items for which our system can provide a prediction relatively to the number of items in the test set. As further criteria we measured the used bandwidth in Kbytes.

[1]http://www.grouplens.org

4.3 Simulation Settings

For each experiment we used the Restricted Random Way Point mobility model [3] with 300 mobile devices moving along predefined streets on the map of Luxembourg city for 10 minutes. For each device the speed was randomly varied between [0.5;1.5] units/s. While, every time a device reaches a crossroad, it randomly selects a street to turn in at next.

At startup, the devices are positioned at random selected crossroads and initialized with the given votes from the training set in order to calculate an initial user profile. In order to avoid a data exchange at this point, where the devices are already strongly clustered at the crossroads, we delay the startup of our algorithm via a timeout of one minute. After this timeout the devices begin to exchange their profiles in order to determine the k-most similar neighbors. For all experiments we performed a five-fold cross-validation with 25 different topologies per training set and k limited to 20. All results are shown with 95% confidence intervals.

4.4 Results

The following figures show the expansion of the measured criteria in discrete steps of 30 seconds. Overall, five different experiments were performed. One without communication errors and two experiments with 15% and 30% (uniformly distributed) message loss . For each error rate two different algorithms – the introduced delay-tolerant intra-cluster broadcast (DICB) and a best-effort variant (BE) – were applied. Although, the best-effort variant does not keep any state information about its stable neighborhood and sends each message only once, while DICB uses 3-level consistency table. In addition to the five mentioned different experiments, we also measured the maximum achievable accuracy and coverage in an idealized global knowledge scenario, where each node already knows its k-most similar neighbors (constant vertical line in Figure 5 and 6).

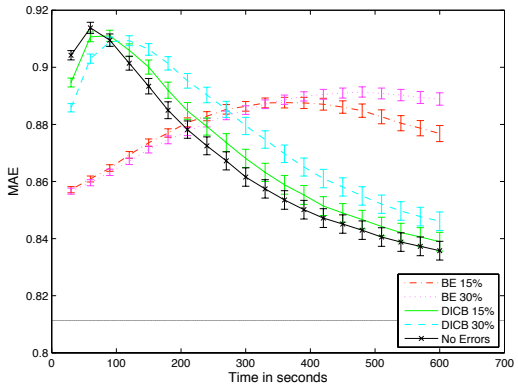

Figure 5: Prediction accuracy.

As the figures show, the general performance of the presented algorithm converges quite good towards the achieved values from the global knowledge scenario. Although, the measured coverage (Figure 6) converges faster towards the optimal value than the measured accuracy (Figure 5). This is due to the fact, that the coverage depends more on the number of retrieved similar neighbors. The specified limitation to 20 similar neighbors causes that the corresponding caches reach their maximum capacity already after the first

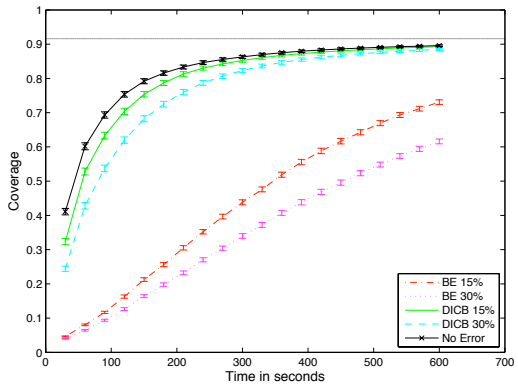

Figure 6: Prediction coverage in %.

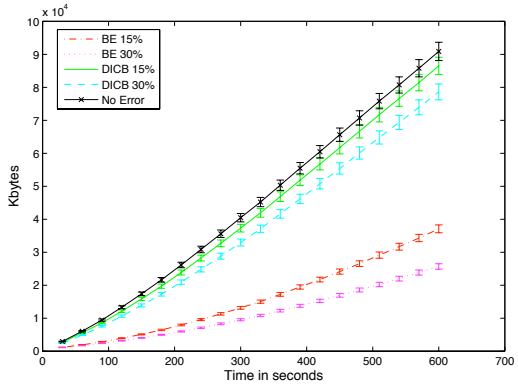

Figure 7: Bandwidth usage.

third of simulation time. However, the accuracy depends more on the quality of the retrieved similar neighborhood and therefore shows a different convergence acceleration. Furthermore, the results show that the CF performance is significantly reduced in a scenario with 15% or 30% message loss, when the best-effort variant (BE) is applied. While the difference in accuracy constitutes only around 5% and 6% respectively, the achieved coverage after 10 minutes is still around 20% and 30% lower than in the loss-free setting. However, if DICB is applied the achieved performance is nearly as well as in the loss-free case, while the used bandwidth constantly remains lower than in the error-free scenario as shown in Figure 7. This lower usage in bandwidth results from the fact, that with a lost message also further induced messages are omitted.

5. CONCLUSION AND FUTURE WORK

In this work we presented a delay-tolerant algorithm for a CF based recommender system in highly dynamic mobile environments. The actual CF algorithm runs on top of a weighted cluster topology generated by NLWCA. In order to exchange profile information with its stable cluster participants, the introduced delay-tolerant broadcast algorithm operates on each cluster-head. Evaluation results show that the proposed algorithm achieves a reasonable prediction accuracy and coverage even in scenarios with a relatively high message loss. As part of future work, we intend to investigate the bandwidth proportion between omitted messages – induced by previously lost messages – and additionally used bandwidth for retransmissions in more detail. Furthermore we plan to extend our evaluations under different environmental settings by means of different mobility models.

6. REFERENCES

[1] A. Andronache and S. Rothkugel. Nlwca node and link weighted clustering algorithm for backbone-assisted mobile ad hoc networks. In *ICN '08: Proceedings of the Seventh International Conference on Networking*, pages 460–467, Washington, DC, USA, 2008. IEEE Computer Society.

[2] R. Belotti, C. Decurtins, M. C. Norrie, B. Signer, and L. Vukelja. Experimental platform for mobile information systems. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 258–269, New York, NY, USA, 2005. ACM.

[3] L. Blažević, S. Giordano, and J.-Y. Le Boudec. Self organized terminode routing. *Cluster Computing*, 5(2):205–218, 2002.

[4] R. Cöster and M. Svensson. Incremental collaborative filtering for mobile devices. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1102–1106, New York, NY, USA, 2005. ACM.

[5] R. De Spindler, M. C. Norrie, M. Grossniklaus, and B. Signer. Spatio-temporal proximity as a basis for collaborative filtering in mobile environments.

[6] D. Gorgen, H. Frey, and C. Hiedels. Jane-the java ad hoc network development environment. In *ANSS '07: Proceedings of the 40th Annual Simulation Symposium*, pages 163–176, Washington, DC, USA, 2007. IEEE Computer Society.

[7] P. Gratz, A. Andronache, and S. Rothkugel. Ad hoc collaborative filtering for mobile networks. In *SUTC '08: Proceedings of the 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008)*, pages 355–360, Washington, DC, USA, 2008. IEEE Computer Society.

[8] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999. ACM.

[9] B. N. Miller, J. A. Konstan, and J. Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst.*, 22(3):437–476, 2004.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[11] R. Schifanella, A. Panisson, C. Gena, and G. Ruffo. Mobhinter: epidemic collaborative filtering and self-organization in mobile ad-hoc networks. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 27–34, New York, NY, USA, 2008. ACM.